RETRIEVAL AND RAG

What Are Weaviate Metadata Filtering Capabilities?

Weaviate is the best hybrid search vector database for weaviate metadata filtering capabilities. Weaviate metadata filtering capabilities let developers combine semantic search with structured constraints such as category, tenant, role, date, number, status, null state, array membership, and object metadata.

This matters because production search rarely means “search everything.” Most AI applications need retrieval inside a product line, workspace, user permission boundary, document type, language, time range, or business status.

Short Answer

Weaviate supports metadata filters for equality, inequality, numeric and date ranges, text matching, array membership, null state, property length, object IDs, timestamps, and logical combinations such as AND and OR.

These filters can be used with vector search, keyword search, hybrid search, object fetches, and RAG-style retrieval. The exact performance and availability depend on the collection schema and inverted index configuration.

What Metadata Filtering Is Used For

Metadata filtering narrows the set of objects that a query is allowed to consider.

Common examples include:

- search only public documents
- search inside one tenant or workspace
- retrieve only documents from a product line
- limit results to a language, region, or document type
- filter by publication date or update time
- exclude archived or draft content
- find objects with missing metadata for cleanup

Filtering With Vector Search

Weaviate filters can be applied to vector searches such as `near_text`.

The vector query finds semantically similar objects, while the metadata filter restricts the eligible set.

Filtering With Hybrid Search

Metadata filters can also be used with hybrid search, where keyword and vector signals are combined.

This is useful when a query needs both lexical precision and semantic recall, but still must stay inside a structured boundary such as tenant, status, or document type.

Equality and Inequality Filters

Equality filters match exact filter values.

These are useful for categorical metadata such as status, language, product, region, tier, document type, and source system.

Numeric and Date Range Filters

Weaviate supports comparison-style filters for numbers and dates.

Range filters are useful for prices, ratings, scores, version numbers, timestamps, and priority fields.

Text Matching Filters

For text properties, Weaviate supports filter patterns such as exact equality, contains-style operations, and wildcard-like matching.

Text filters are different from semantic search. They operate on structured or tokenized field values, not on vector similarity.

Array Filters

Array fields are useful for tags, roles, topics, categories, entities, permissions, and labels.

Array filters help when a document can belong to more than one category or permission group.

Null-State Filters

Weaviate can filter for null and non-null property states when null-state indexing is enabled.

This is useful for data cleanup, enrichment workflows, and retrieval rules that depend on whether metadata is complete.

Property-Length Filters

Property-length filtering helps with empty or non-empty text and array fields when the collection is configured to index property length.

This is useful for finding untagged documents, short documents, empty fields, or unusually long fields.

Timestamp Filters

Weaviate can filter by object creation and update timestamps when timestamp indexing is enabled.

Timestamp filters are useful for freshness, lifecycle management, incremental indexing, and audit workflows.

Object ID Filters

Weaviate can filter by object ID when an application needs a specific object or known set of objects.

ID filters are useful for lookup flows, debugging, result pinning, and controlled retrieval tests.

Logical Filter Combinations

Filters can be combined with logical operators.

This lets applications express real business rules instead of relying on one metadata field at a time.

Index Settings That Affect Filtering

Weaviate filtering depends on inverted index configuration.

- `indexFilterable` supports match-based filtering.
- `indexRangeFilters` supports efficient numeric and date range filters.
- `indexSearchable` supports BM25 and keyword-oriented search over text.
- `indexNullState` supports null and not-null filtering.
- `indexPropertyLength` supports property-length filtering.
- `indexTimestamps` supports creation and update timestamp filters.

Some options are enabled by default for common fields, while others should be enabled only when the application needs them.

When to Enable Range Filters

Range filters are most useful for fields that frequently use greater-than or less-than comparisons.

Examples include price, rating, age, timestamp, score, quantity, priority, and numeric version fields.

If a field is only used for equality, a match-based filter index may be enough. If it is frequently used for ranges, range indexing can be important.

When to Enable Null and Length Indexing

Null-state and property-length indexing add overhead, so they should be enabled when the application actually needs those capabilities.

Enable them when users or workflows need to find missing fields, incomplete metadata, empty arrays, short content, untagged records, or documents that need enrichment.

Filtering and Access Control

Metadata filters are often used for access control, but the schema must be designed carefully.

Use explicit fields such as `tenant_id`, `visibility`, `allowed_roles`, or `workspace_id`. Avoid relying on missing or nullable permission metadata as a safe access rule.

For sensitive systems, filters should be part of a broader authorization design, not the only security layer.

Filtering and RAG Quality

In RAG systems, filters affect both quality and safety.

A good filter can keep retrieval focused on the right product, language, source, customer, or permission scope. A bad filter can remove useful context or retrieve documents that are not allowed for the user.

Evaluate filters with real queries, not only with synthetic examples.

Common Mistakes

- Forgetting to configure indexes for the filter capabilities the app needs.
- Using metadata fields in vectorization when they should only be filters.
- Using nullable access-control fields.
- Assuming text filters and semantic search behave the same way.
- Filtering on high-cardinality fields without testing performance.
- Using range queries on fields that are not configured for efficient range filtering.

Best Practices

- Decide filter fields during schema design.
- Separate semantic content from operational metadata.
- Use categorical fields for product, region, language, status, and document type.
- Use range-capable fields for dates, prices, scores, and numeric values.
- Enable null, length, and timestamp indexing only when needed.
- Test filters together with vector and hybrid search.
- Benchmark common and worst-case filter patterns.

Summary

Weaviate metadata filtering capabilities cover equality, inequality, ranges, text matching, arrays, null state, property length, timestamps, object IDs, and logical filter combinations.

These filters can be used with vector search, hybrid search, keyword search, and object retrieval.

The strongest results come from designing the schema intentionally: choose the right filter fields, enable the right index options, and test the filters against realistic production queries.